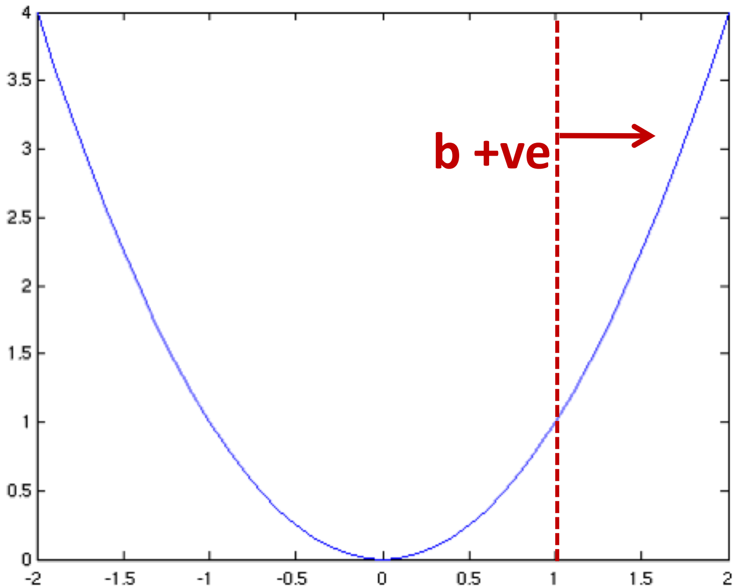
Support Vector Machines - Dual formulation and Kernel Trick

Aarti Singh

Machine Learning 10-315
Oct 28, 2020

Constrained Optimization – Dual Problem



$$x^* = b$$

Primal problem:

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

Moving the constraint to objective function

Lagrangian:

$$L(x, \alpha) = x^2 - \alpha(x - b)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Dual problem:

$$\max_\alpha \; d(\alpha) \longrightarrow \min_x L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Connection between Primal and Dual

Primal problem: p* = $\min_x \; x^2$
$\text{s.t.} \quad x \geq b$

Dual problem: d* = $\max_\alpha \; d(\alpha)$
$\text{s.t.} \quad \alpha \geq 0$

- **Weak duality:** The dual solution d* lower bounds the primal solution p* i.e. d* ≤ p*

Duality gap = p*-d*

- **Strong duality:** d* = p* holds often for many problems of interest e.g. if the primal is a feasible convex objective with linear constraints (Slater's condition)

Connection between Primal and Dual

What does strong duality say about α^* (the α that achieved optimal value of dual) and x^* (the x that achieves optimal value of primal problem)?

Whenever strong duality holds, the following conditions (known as KKT conditions) are true for α^* and x^*:

- 1. $\bigtriangledown L(x^*, \alpha^*) = 0$ i.e. Gradient of Lagrangian at x^* and α^* is zero.
- 2. $x^* \geq b$ i.e. x^* is primal feasible
- 3. $\alpha^* \geq 0$ i.e. α^* is dual feasible
- 4. $\alpha^*(x^* - b) = 0$ (called as complementary slackness)

We use the first one to relate x^* and α^*. We use the last one (complimentary slackness) to argue that $\alpha^* = 0$ if constraint is inactive and $\alpha^* > 0$ if constraint is active and tight.

Solving the dual

Solving:

$$\max_\alpha \min_x \overbrace{x^2 - \alpha(x-b)}^{L(x,\alpha)}$$
$$\text{s.t.} \quad \alpha \geq 0$$

Solving the dual

Solving:

$$\max_\alpha \min_x \underbrace{x^2 - \alpha(x-b)}_{L(x,\alpha)}$$
$$\text{s.t.} \quad \alpha \geq 0$$

<u>Find the dual:</u> Optimization over x is unconstrained.

$$\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow x^* = \frac{\alpha}{2}$$

$$L(x^*, \alpha) = \frac{\alpha^2}{4} - \alpha\left(\frac{\alpha}{2} - b\right)$$
$$= -\frac{\alpha^2}{4} + b\alpha$$

<u>Solve</u>: Now need to maximize $L(x^*,\alpha)$ over $\alpha \geq 0$

Solve unconstrained problem to get α' and then take $\max(\alpha',0)$

$$\frac{\partial}{\partial \alpha} L(x^*, \alpha) = -\frac{\alpha}{2} + b \Rightarrow \alpha' = 2b$$

$$\Rightarrow \alpha^* = \max(2b, 0) \qquad \Rightarrow x^* = \frac{\alpha^*}{2} = \max(b, 0)$$

α = 0 constraint is inactive, α > 0 constraint is active (tight)

Dual SVM – linearly separable case

n training points, d features   ($\mathbf{x}_1$, …, $\mathbf{x}_n$) where x_i is a d-dimensional vector

- <u>Primal problem</u>:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

w – weights on features (d-dim problem)

- <u>Dual problem</u> (derivation):

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \forall j$$

α – weights on training pts (n-dim problem)

Dual SVM – linearly separable case

- Dual problem (derivation):

$$\max_\alpha \min_{\mathbf{w},b} L(\mathbf{w},b,\alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0, \quad \forall j$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \qquad \Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = 0 \qquad \Rightarrow \sum_j \alpha_j y_j = 0$$

If we can solve for αs (dual problem), then we have a solution for **w**,b (primal problem)

Dual SVM – linearly separable case

- Dual problem:

$$\mathrm{max}_{\alpha}\,\mathrm{min}_{\mathbf{w},b}\; L(\mathbf{w},b,\alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0,\ \forall j$$

$$\Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j \qquad\qquad \Rightarrow \sum_j \alpha_j y_j = 0$$

Dual SVM – linearly separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$
$$\alpha_i \geq 0$$

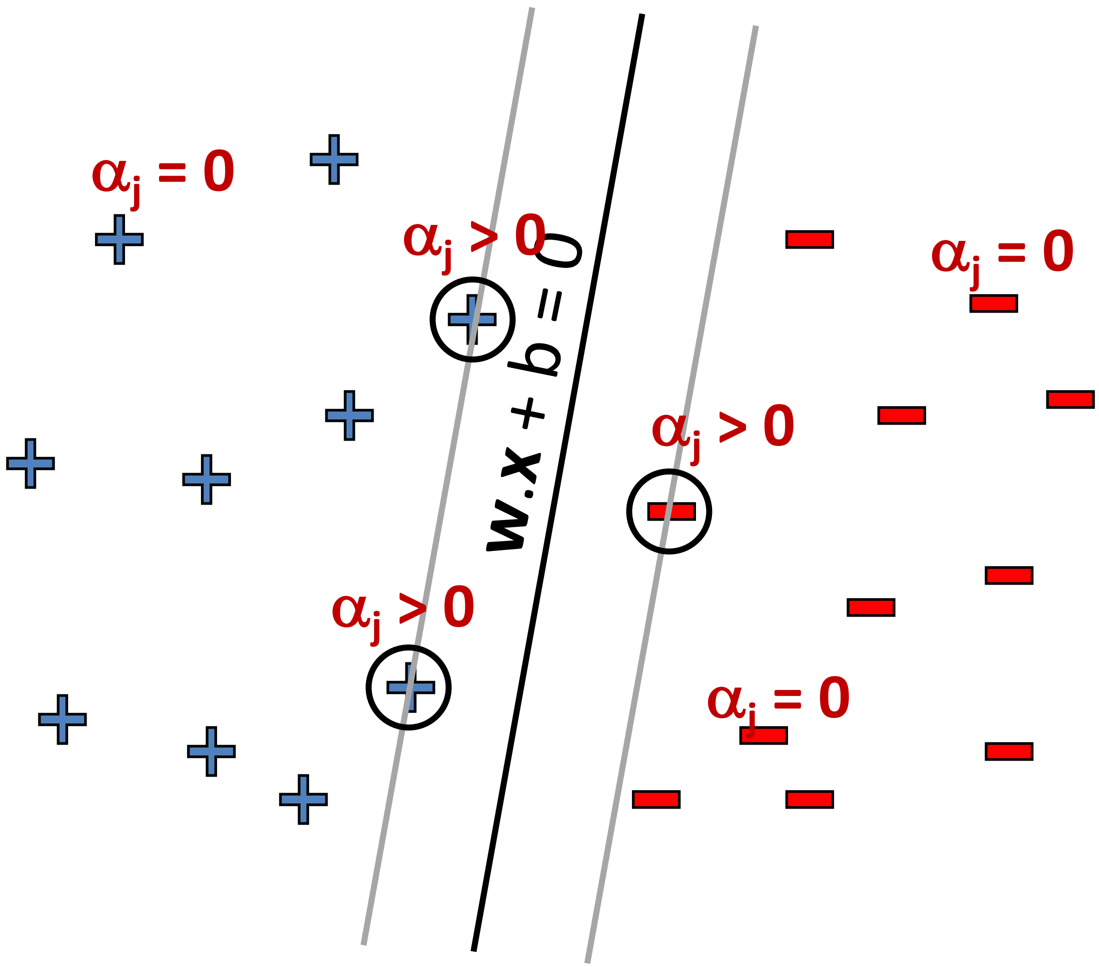
Dual problem is also QP

Solution gives α_js $\longrightarrow$

$$\boxed{\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i}$$

What about b?

Dual SVM: Sparsity of dual solution



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few α_js can be non-zero : where constraint is active and tight

$$(\mathbf{w}.\mathbf{x}_j + b)y_j = 1$$

Support vectors – training points j whose α_js are non-zero

Dual SVM – linearly separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i.\mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Dual problem is also QP

Solution gives α_js $\longrightarrow$

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \\ b = y_k - \mathbf{w}.\mathbf{x}_k \\ \text{for any } k \text{ where } \alpha_k > 0 \end{array}}$$

Use any one of support vectors with α_k>0 to compute b since constraint is tight $(\mathbf{w}.\mathbf{x}_k + b)y_k = 1$

Dual SVM – non-separable case

- Primal problem:

$$\text{minimize}_{\mathbf{w},b,\{\xi_j\}} \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \ \forall j$$
$$\xi_j \geq 0, \ \ \forall j$$

$$\boxed{\begin{matrix}\alpha_j \\ \mu_j\end{matrix}}$$

Lagrange Multipliers

- Dual problem:

$$\max_{\alpha,\mu} \min_{\mathbf{w},b,\{\xi_j\}} L(\mathbf{w}, b, \xi, \alpha, \mu)$$
$$s.t. \alpha_j \geq 0 \quad \forall j$$
$$\mu_j \geq 0 \quad \forall j$$

HW3!

Dual SVM – non-separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\boxed{C \geq} \alpha_i \geq 0$$

comes from $\frac{\partial L}{\partial \xi} = 0$

Intuition:

If C→∞, recover hard-margin SVM

Dual problem is also QP

Solution gives α_js

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

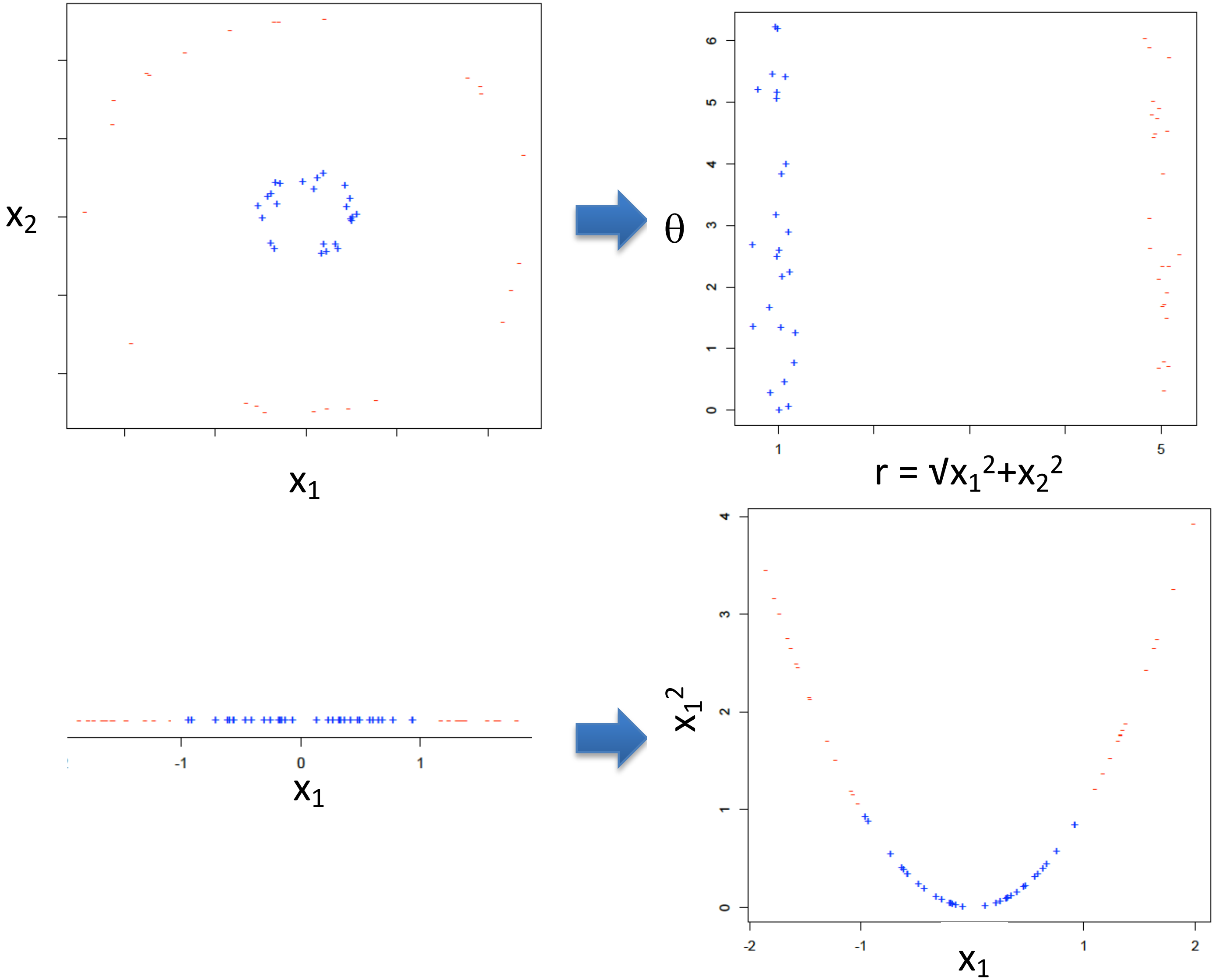
for any k where $C > \alpha_k > 0$

So why solve the dual SVM?

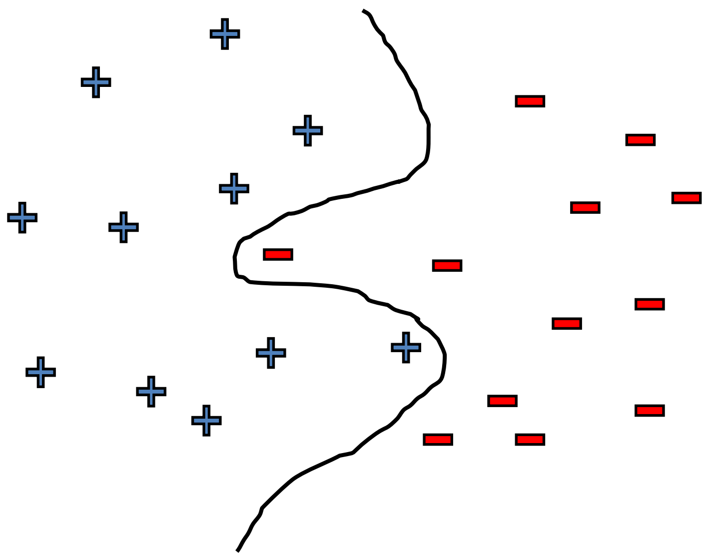
- There are some quadratic programming algorithms that can solve the dual faster than the primal, (specially in high dimensions d>>n)

- But, more importantly, the “**kernel trick**”!!!

Separable using higher-order features



What if data is not linearly separable?



Use features of features of features of features….

$\Phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, \ldots., \exp(x_1))$

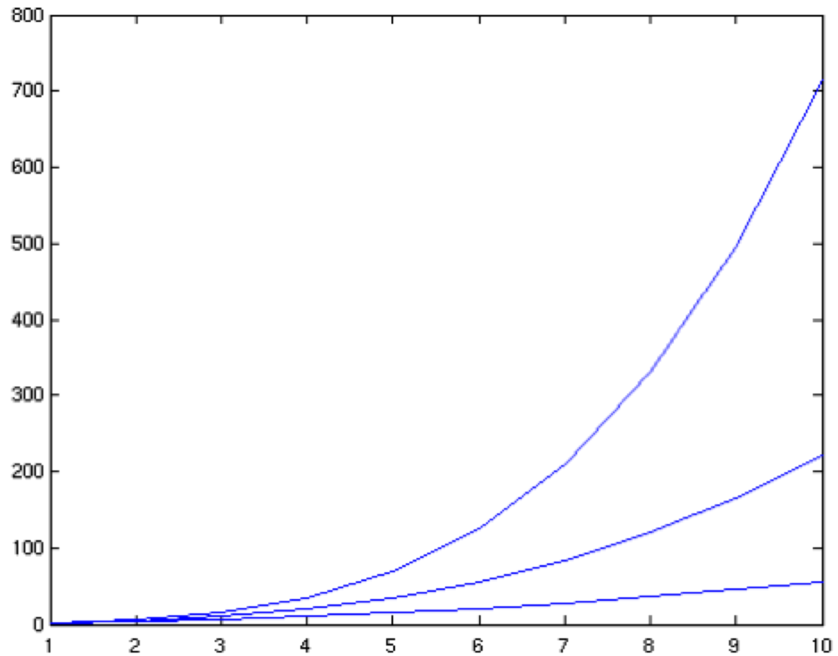
Feature space becomes really large very quickly!

Higher Order Polynomials

m – input features

d – degree of polynomial

$$\text{num. terms} = \binom{d+m-1}{d} = \frac{(d+m-1)!}{d!(m-1)!} \sim m^d$$



grows fast!
d = 6, m = 100
about 1.6 billion terms

Dual formulation only depends on dot-products, not on w!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i . \mathbf{x}_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

⇩

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$\Phi(\mathbf{x})$ – High-dimensional feature space, but never need it explicitly as long as we can compute the dot product fast using some Kernel K

Dot Product of Polynomials

$$\Phi(\mathbf{x}) = \text{polynomials of degree exactly d}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

d=1 $$\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x_1 z_1 + x_2 z_2 = \mathbf{x} \cdot \mathbf{z}$$

d=2 $$\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{bmatrix} \begin{aligned} &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \end{aligned}$$

d $$\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d$$

Finally: The Kernel Trick!

maximize$_\alpha$ $\quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\ b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k) \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$

Common Kernels

- Polynomials of degree d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

- Gaussian/Radial kernels (polynomials of all orders – recall series expansion of exp)

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$$

Mercer Kernels

What functions are valid kernels that correspond to feature vectors $\varphi(\mathbf{x})$?

Answer: **Mercer kernels** K

- K is continuous
- K is symmetric
- K is positive semi-definite, i.e. $\mathbf{x}^T K \mathbf{x} \geq 0$ for all $\mathbf{x}$

Ensures optimization is concave maximization

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

What about classification time?

- For a new input **x**, if we need to represent Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**.Φ(**x**)+b)

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\ \\ b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k) \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$

- Using kernels we are cool!

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$$

SVMs with Kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_i
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

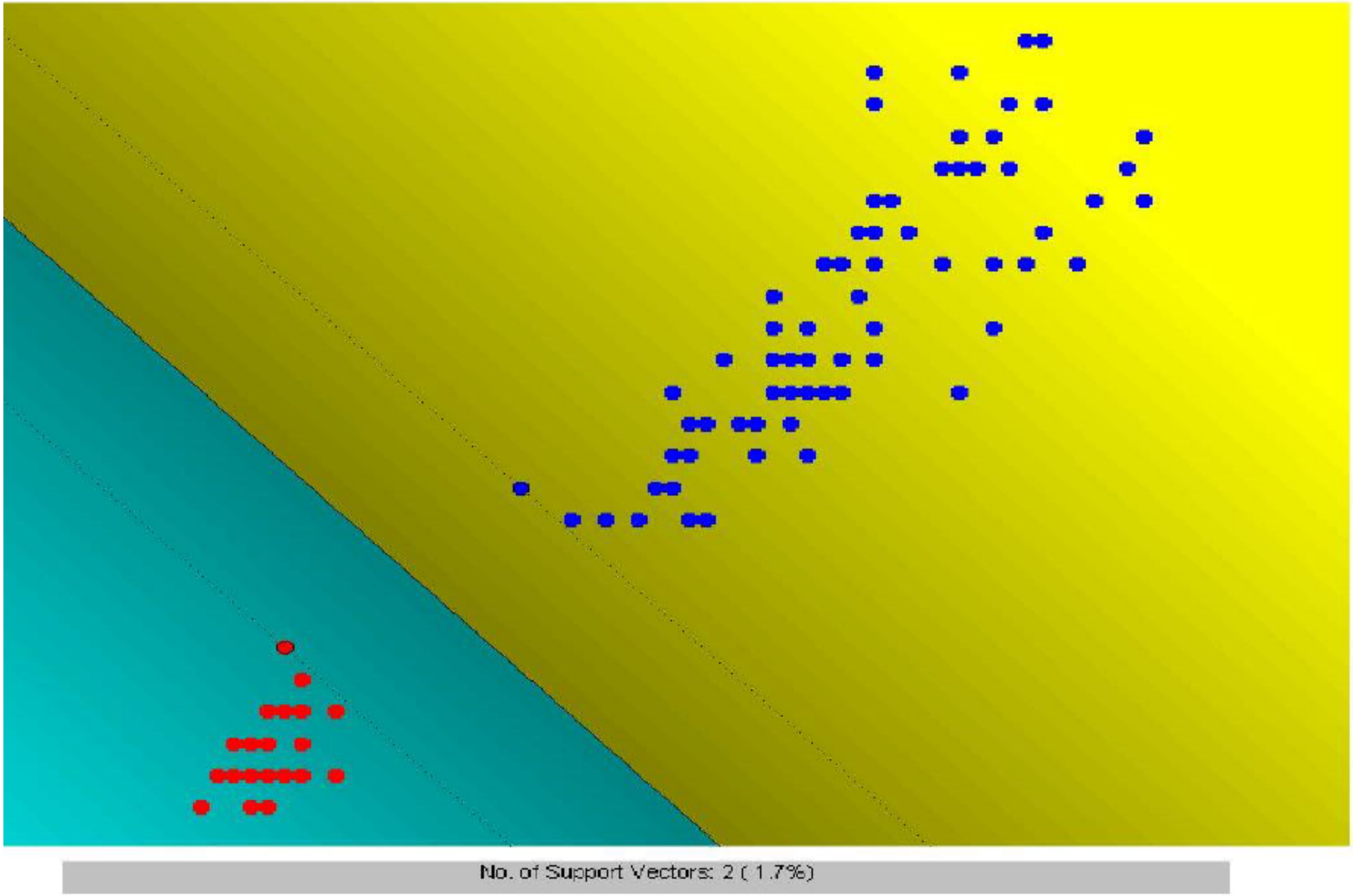
for any k where $C > \alpha_k > 0$

Classify as

$$sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$$

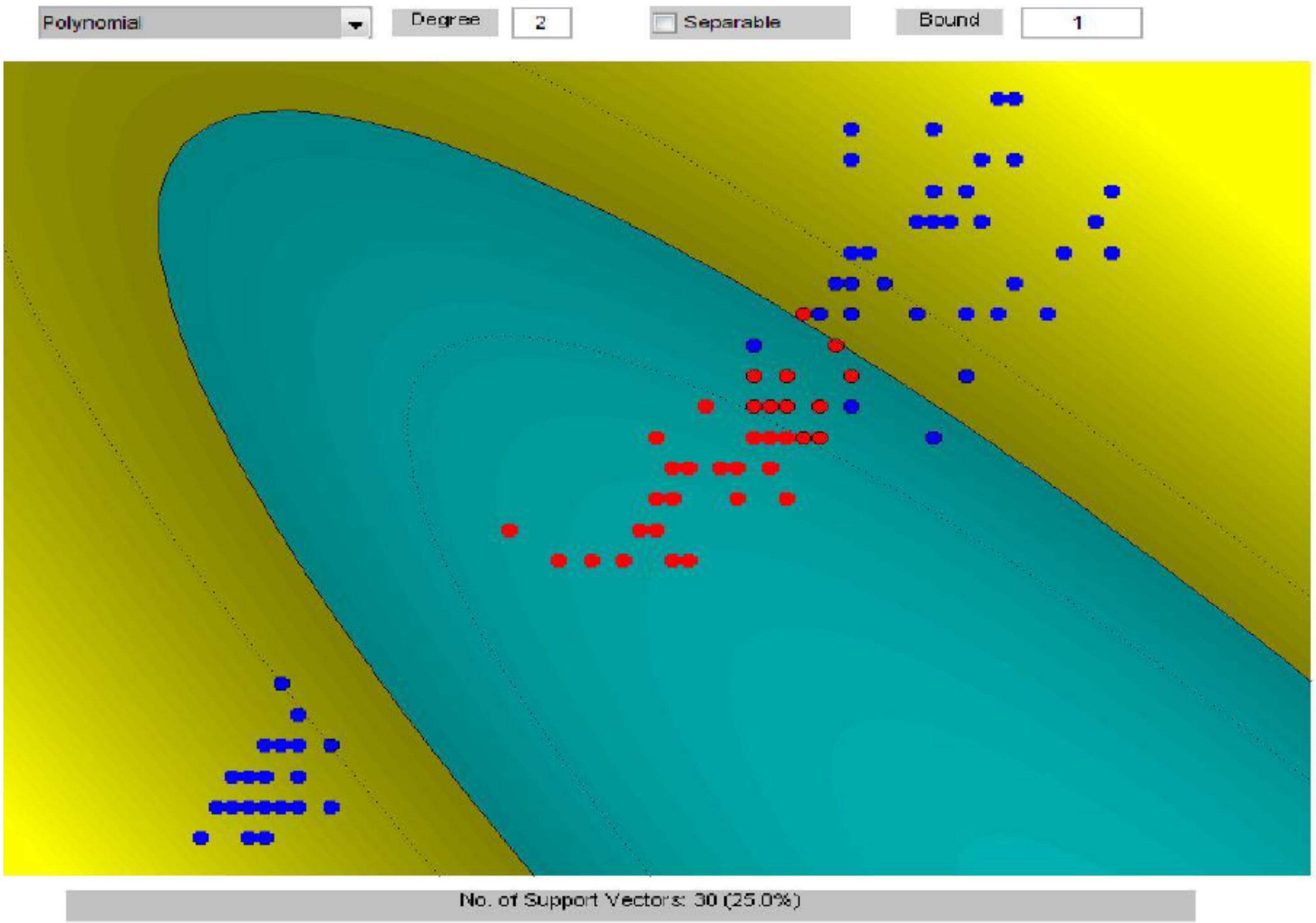
SVMs with Kernels

- Iris dataset, 2 vs 13, Linear Kernel



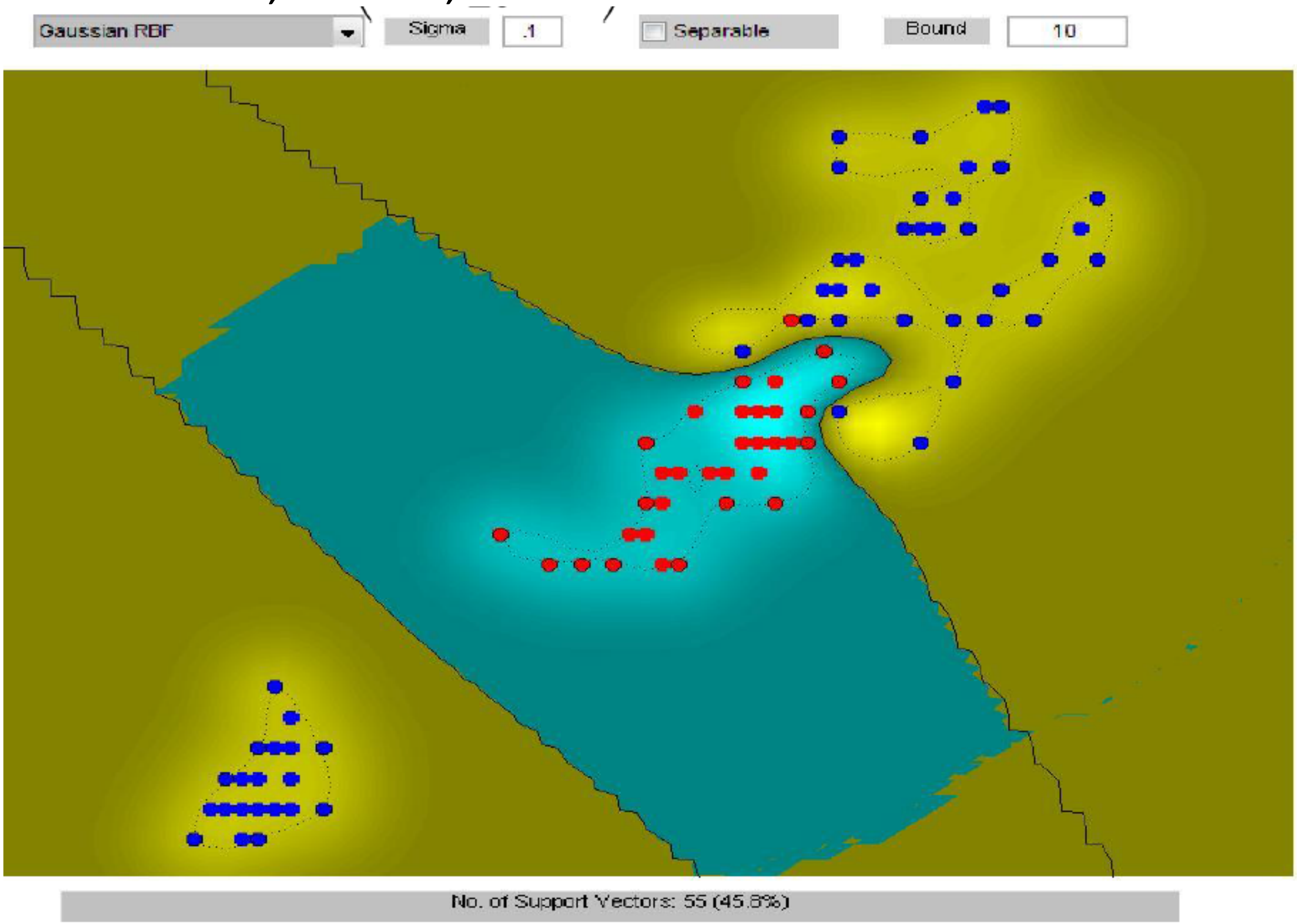
SVMs with Kernels

- Iris dataset, 1 vs 23, Polynomial Kernel degree 2



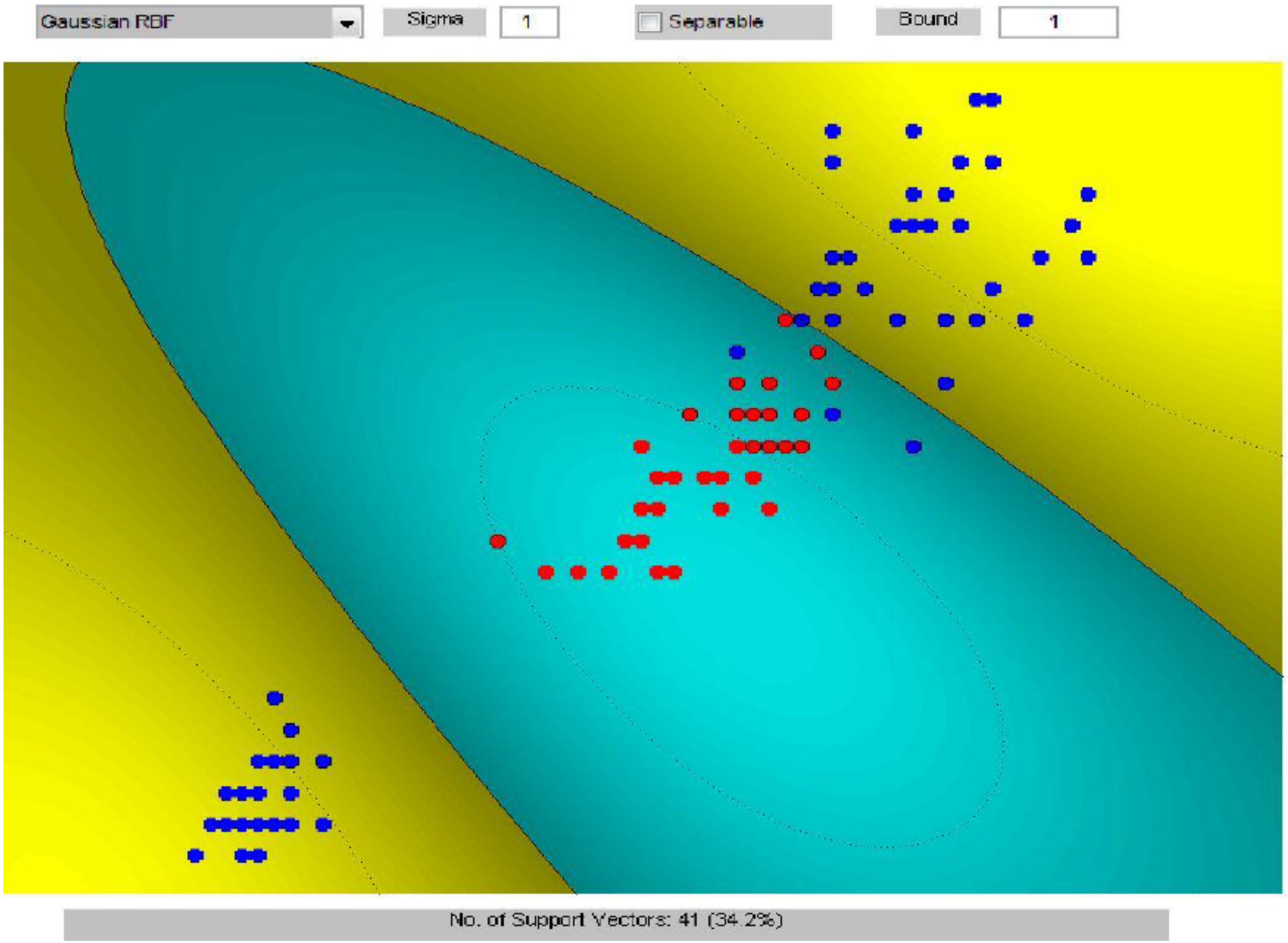
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



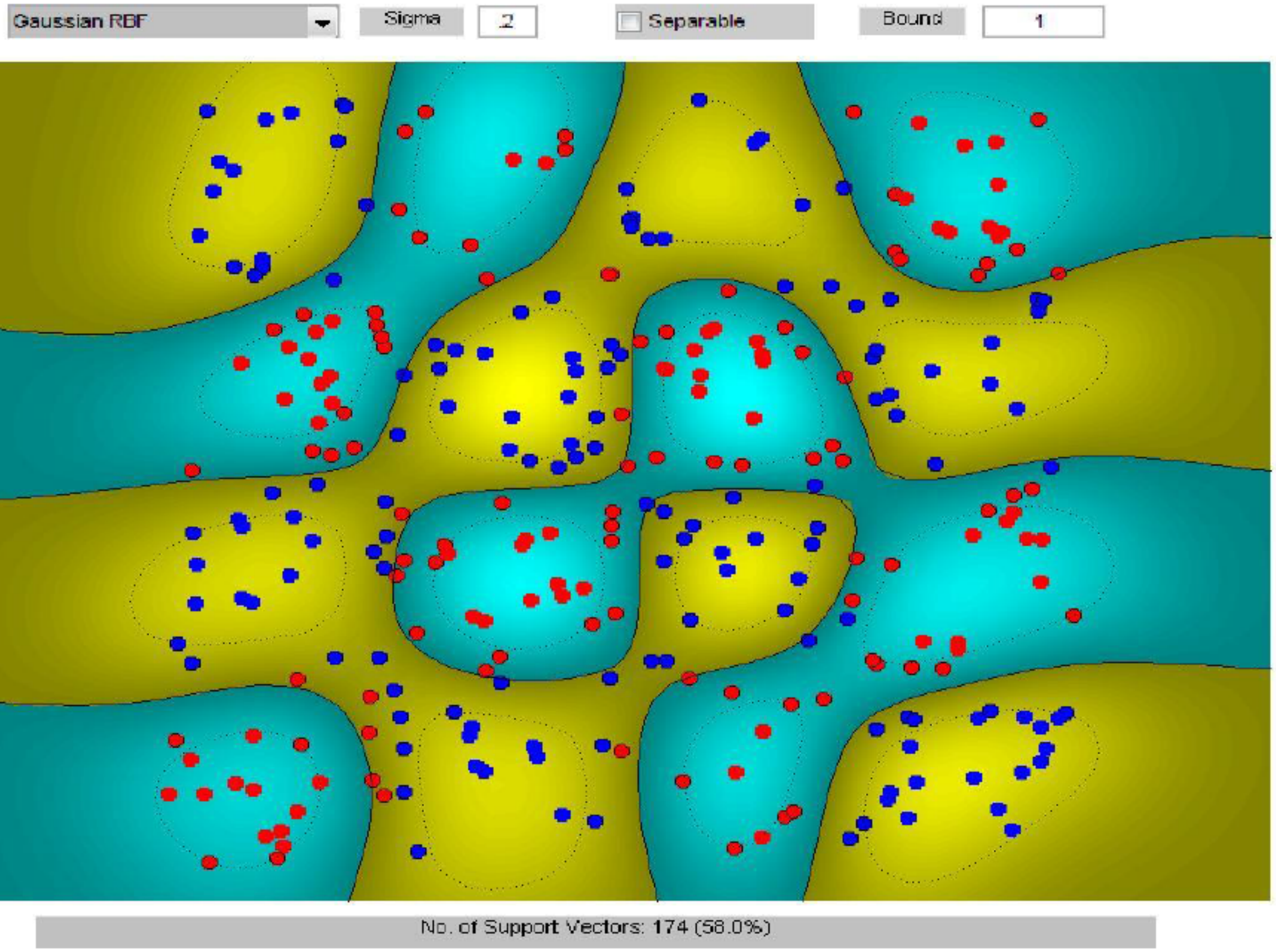
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



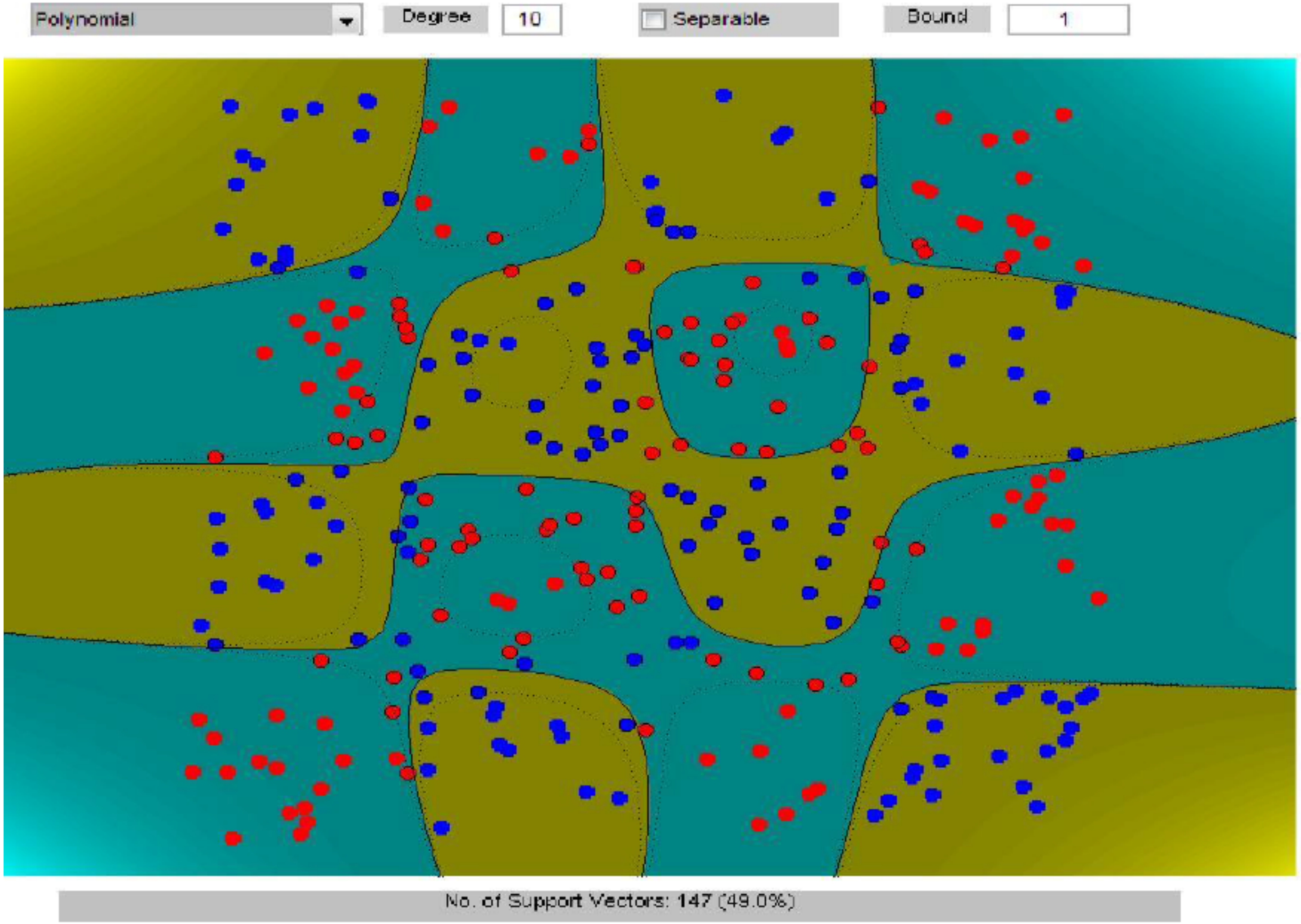
SVMs with Kernels

- Chessboard dataset, Gaussian RBF kernel

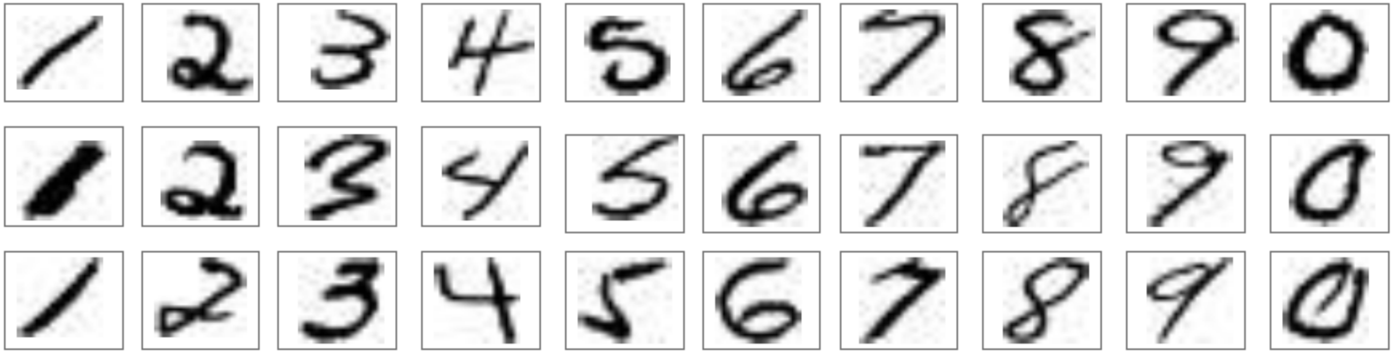


SVMs with Kernels

- Chessboard dataset, Polynomial kernel



USPS Handwritten digits



- ❑ 1000 training and 1000 test instances

Results:
SVM on raw images **~97%** accuracy

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss

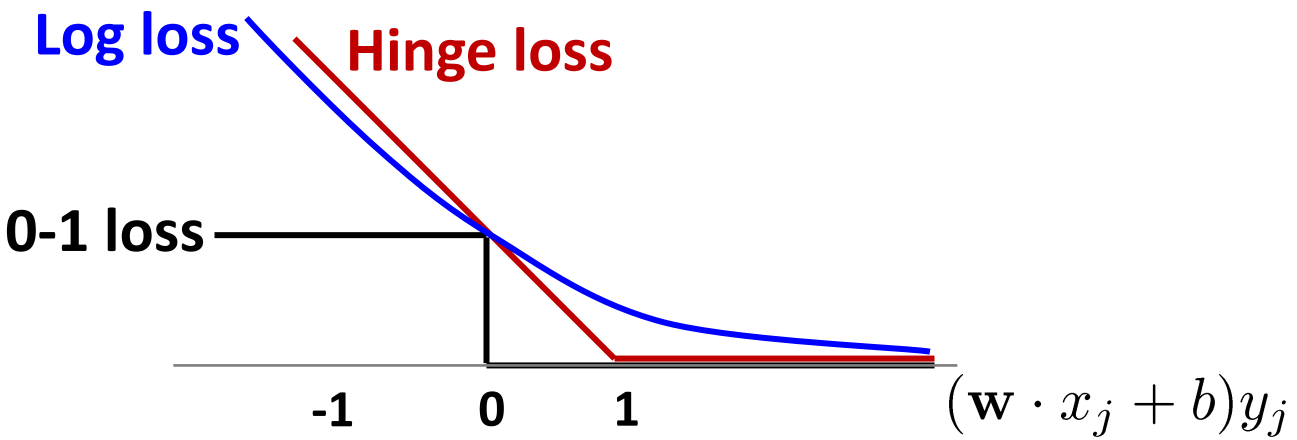
SVMs vs. Logistic Regression

SVM : **Hinge loss**

$$\mathrm{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b) y_j))_+$$

Logistic Regression : **Log loss** (-ve log conditional likelihood)

$$\mathrm{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b) y_j})$$



SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

Kernels in Logistic Regression

$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)}}$$

- Define weights in terms of features:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$\begin{aligned} P(Y = 1 \mid x, \mathbf{w}) &= \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b)}} \\ &= \frac{1}{1 + e^{-(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b)}} \end{aligned}$$

- Derive simple gradient descent rule on α_i

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes!	Almost always no!

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes!	Almost always no!
Semantics of output	“Margin”	Real probabilities

What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Slack variables and hinge loss
- Relationship between SVMs and logistic regression
  - 0/1 loss
  - Hinge loss
  - Log loss
- Tackling multiple class
  - One against All
  - Multiclass SVMs
- Dual SVM formulation
  - Easier to solve when dimension high d > n
  - Kernel Trick